DOCUMENT RESUME

ED 311 739 FL 018 189

TITLE	The Multi-Language Multi-Media Courseware Authoring and Delivering System. Final Report.
INSTITUTION	Comptek Co., Springfield, NJ.
SPONS AGENCY	Office of Educational Research and Improvement (ED), Washington, DC.
PUB DATE	27 Aug 87
CONTRACT	400-85-1012
NOTE	47p.
PUB TYPE	Reports - Descriptive (141)
EDRS PRICE	MF01/PC02 Plus Postage.
DESCRIPTORS	Audio Equipment; *Authoring Aids (Programing); *Chinese; *Computer Assisted Instruction; *Courseware; *English; Higher Education; Ideography; *Japanese; Material Development; Microcomputers; *Multimedia Instruction; Program Descriptions; Second Language Instruction; Uncommonly Taught Languages

ABSTRACT

The development of a multi-media, multi-language authoring and student use system for second language instruction is described. The system uses a microcomputer and monitor, off-shelf voice peripheral device, microphone, speaker, and over 40,000 lines of courseware. It allows an author or teacher to compose courseware and present it to the students. The courseware can control the sequence of presentation depending on the student's response and data accumulated during the course. The author's interface with the system is menu-driven and screen-oriented. Courseware can be prepared and presented in English, Chinese, or Japanese and in text, graphics, or voice. The system has access to over 3,000 Chinese and Japanese Kanji characters. It was field tested by professors for first-year Chinese courses for college students. (MSE)

Final Report Summary

Title: The Multi-Language Multi-Media Courseware Authoring and Delivering System

Department of Education, SBIR Phase-II, Contract No. 400-85-1012
RFP No. NIE-R-85-0005
RFP Topic: Simplifying and Improving the Creation of Software

Prepared By: Comptek Company
P. O. Box 245, Springfield, N.J. 07081
Tang Mao, Managers, Government Contract
Date: August 27, 1987

Technical Abstract: Comptek completed a prototype of a Multi-language, Multi-media, Authoring and Student (MMAS) system. It is composed of an IBM-PC or compatible with color or graphics monitor, an off-shelf voice peripheral device, a microphone, a speaker, and over forty thousand lines of software developed during Phase-II. As a courseware authoring and delivering system, it allows an author or teacher to compose a courseware and presents the courseware to the student(s). The courseware can control the sequence of presentation depending on the response from the student and accumulated data during the course. The author's interface with the MMAS is menu driven and screen oriented. The courseware can be prepared and presented in multiple languages: English, Chinese, or Japanese, and in multiple media: text, graphics, or voice. The system has access to more than 3,000 Chinese and Japanese Kanji characters. The system was field tested by professors for first year Chinese courses for college students at Boston. Several demo and real coursewares were constructed.

This report presents the MMAS, the work performed, the results achieved, their evaluation, the difficulties encountered and resolved, and future enhancements foreseen.

Key Words: Courseware, Computer Aided Instruction (CAI), Chinese, Japanese, Authoring System, Language Instruction.

The Results and Potential Applications: The result of the Phase-II is a working system that can be used to prepare and present courseware in three languages in the forms of screen monitor and voice messages. Besides as a normal courseware system, it is especially suitable for language instructions in teaching students speaking one language about the other two languages.

TABLE OF CONTENTS

1. Phase-II Technical Objectives

As stated in Phase-II Proposal Chapter 3 (page 3-1), the technical objectives of the project are: (a) to carry out a complete implementation of the MMAS prototype, and (b) to demonstrate in a satisfactory fashion certain desirable capabilities.

More specifically, We expect the MMAS system to be able to:

1. Provide graphic, text, and voice capabilities,

   so that the courseware author can create courseware that presents the educational material with color graphs, alpha-numeric text in various fonts and sizes, along with the voice. The voice in our system is not generated by a voice synthesizer, but is the human sound recorded previously.

2. Capable of preparing and presenting in multiple languages,

   so that the courseware can be prepared and presented in many languages, intermixed with each other. The goal for the Phase-II project is Chinese and Japanese, besides English. The capability to extend the system to include other languages, however, is important and would be preserved.

3. Friendly to the author, yet powerful enough,

   so that the author with little knowledge about computer can create, or compose, the courseware within a much shorter period of time than using existing systems.

4. Affordable,

   so that many institutions presently can not afford the high cost of mainframe, mini-computer, and special peripherals for the CAI applications can have the luxury of not only educating their students, but also providing their teachers a means to create their own coureware on line.

5. Portable to other delivering systems,

   so that a courseware created for a particular delivering system can be used on another delivering system with insignificant effort.

6. Portable to other authoring systems,

so that the authoring system can be moved to another family of computers with a significantly less effort than developing a new authoring system for the new computers.

2. Work Performed

This chapter describes the work performed during the Phase-II Project. In summary, we first established the system architecture and the hardware / software foundations. Based on these, we adapted Chinese and Japanese databases to provide thousands of characters needed in using the two languages. To knit all parts together, we developed a large amount of C programming language code to provide the functions of the MMAS, both the authoring system and the delivering system. We also completed a user's manual, besides internal design documents, for the system. To further assist the use of the system, we developed a demo courseware, and a real courseware that teaches Chinese for English speaking students during the field trial. Finally, we held a four-month field trial at Boston, where Professor Hu and her colleagues tried and evaluated the system.

Each of the work activities are further discussed in the following Sections. (It should be noted that the work items described are only approximately in the chronological order. Many activities overlapped with one another. Several, such as the software development, lasted for most of the project period.)

2.1 System Architecture and Foundations

The basic system architecture of the MMAS was researched during the Phase-I Project, and was presented in Sections 2.3 and 2.4 of the Phase-II Proposal (January 22, 1985). Based on that architecture, the MMAS was configured to include one IBM-PC with 256 kilo-byte memory, one Dialogic voice peripheral card, a microphone (needed for the Authoring System only) and a speaker. The software foundation, upon which our application software built on, includes the DOS operating system, the Lattice-C library, the VDI graphics library, and the voice device driver/ library. All components were leased or purchased, integrated together, and a fair amount of efforts were spent to test each component for the adequateness for our application. Several software programs were developed for the testing purpose.

2.2 Databases

Unlike the Romanic languages, the two target languages in the MMAS, Chinese and Japanese, have thousands of basic characters, each distinct. We decided to use a database to store the patterns of those characters and to retrieve the characters when the courseware calls for a display. An alternative to the database approach is to use special

language hardware to provide the characters. The database approach was chosen because it is more flexible (new characters can be added when necessary) and needs no special hardware (thus costs less).

The Chinese database was acquired from Taiwan, and the Japanese Kanji character database from Canada. With the raw databases in hand, we spent much effort to adapt them into the MMAS system. The original Chinese database was on a tape, and the original Japanese database was in the Apple-II disk file format. For each language, we implemented software utilities to select and retrieve characters from the source, entered proper selection sequence (Pin-yin for Chinese and Japanese phonetic symbols for Japanese), built the database, and developed software library interfaces that were invoked in the MMAS system. The different disk file format for the Japanese database required extra effort: it was first read and translated into an Apple McIntosh, shipped to a Unix system, and finally downloaded to the IBM PC for the MMAS application.

Currently the MMAS Chinese database has 3,000 characters, i.e. it can display 3,000 distinct Chinese characters at the discretion of the courseware. This capacity covers about 90% of Chinese characters used in the daily newspapers, sufficient for languages instructions. The MMAS Japanese database also has about 3,000 Kanji characters, of which about 100 had their selection sequences entered into the database and are accessible. The capacity of the accessible Japanese characters is adequate for demo purpose.

To be able to compose new characters, a software tool was developed. It was used to create the fifty Japanese phonetic symbols, and can also be used to add new characters into the Chinese or the Japanese databases. With this flexibility, the database sizes are not limited to the three thousand characters as originally acquired. We can add new characters whenever there is a need. The tool is quite convenient to use, takes only about 2 minutes to enter the pattern of a new character.

In MMAS, the selection sequence, or input method, for Chinese is the Pin-Yin method. The selection sequences for all Chinese characters are specified in the XinHua Zidian (Dictionary), 1984, published by Commerce Book Store, Beijin, China. It is one of the popular input methods for Chinese, and is used in the Chinese courses of foreign language departments in many U.S. universities.

The input method in MMAS for Japanese is the Japanese-phonetic- symbol method, in which each Kanji character is

represented by a sequence of the fifty basic phonetic tokens. All selection sequences of Kanji characters are used in Fujitsu OASIS system and specified in "IBM5550 Dictionary," published by IBM Japan. Like in the Pin-Yin method, the MMAS resolves conflicts (two or more characters having the same sequence) in the phonetic selection method by displaying all conflict characters and prompting the author for another selection.

2.3 Software Development

Software development is the major effort in the Phase-II Project. Over a period of one and a half year, there were total over 40,000 lines of C code developed and tested. The major components and their capabilities are described as follows. A detailed discussion with examples about the usage of each component can be found in the MMAS User's Manual, included as Appendix C.

2.3.1 Course Organizer The Course Organizer enables the author to create the courseware. With its screen oriented, menu driven interface, an author can specify the order to display a screen or to output a voice message. With its logic features, the author can also specify the conditions that alter the presentation sequence. For instance, the author can instruct the courseware to go through some special materials if a student inputs an incorrect answer. Or the author can have the courseware output a complementary voice message when a student enters the fifth consecutive correct answer.

2.3.2 Compiler After the courseware was created via the Organizer, the author invokes the Compiler, also called the Translator in the previous interim reports, to translate the courseware into machine readable form, which the Rehearser understands.

2.3.3 Rehearser The Rehearser is by itself a complete courseware delivery system. Equipped with capabilities to understand the courseware logics, to access Chinese/Japanese database, to display the characters, text, and graphics, to read from the keyboard, and to output to the speaker, the Rehearser can process the courseware and perform according to the courseware's instructions and student inputs. The Rehearser has also trouble-shooting features built in that can be used to diagnose the logics of the courseware while being developed.

2.3.4 Graphics Editor The Graphics Editor, also called the screen editor, can be used to compose the screens used in the courseware. The screens can include a combination of

text, which may be alphabetic, numeric, or Chinese/Japanese, graphics, which may be lines, blocks, marked areas, or other VDI graphics components. The graphics can be in any color allowed by the monitor, of several sizes, and of other attributes as specified in the VDI standard.

With the menu driven interface, even an inexperienced person can learn the use of the Graphics Editor quickly. Typically a fancy screen can be drawn in a matter of minutes. An example is a staff member's seven year old son, who learned the the Editor in one half hour, and created many interesting screens, some of which were included in our demo courseware.

2.3.5 Voice Editor The Voice Editor allows the editing of voice messages for the courseware. The author can speak to the microphone, record the message as a disk file, and later name the file to be output to a speaker at an appropriate time as decided by the courseware. It uses the voice driver to interface with the voice device, the speaker, and the microphone. Other voice library routines were also developed that enabled the Organizer to record the voice message when the author creates the courseware, and the Rehearser to output the voice messages. The voice Editor is also considered friendly. With analogy to a tape recorder, it is generally considered easy to use.

2.4 Documentation

During the Project, numerous internal documents about system design, testing results, and field testing planning were produced. The major document that was produced for the users of the MMAS is "The MMAS Version 1.0 User's Manual," which was intended for the installation and the operation of the system. It is included in its entirety as Appendix C.

Further, as a menu-driven system, the MMAS software has a number of help menus. A help menu provides a briefed explanation about available options at any given time during an authoring session. The menus are useful as a reminder for those who used the MMAS before or who have some knowledge about the system.

2.5 Courseware

During the development of the MMAS, several testing coursewares were developed to exercise and diagnose the various parts of the software. Before the start of the field trial, a simple demo courseware was developed to demonstrate the features of the MMAS and its use in Chinese language instructions. During the field trial at Boston,

Consultant Hu and Comptek staff jointly developed a courseware intended for the first year Chinese language course. A sample of that courseware, which uses the MMAS' voice capability to teach students about "sound discrimination," is included as Appendix B.

2.6 Field Trial

From February to June 1987, a field trial was held at East Asia Civilization and Language Department at Harvard University, Boston. The purpose of the trial was to verify and measure several important properties of the MMAS system: functional correctness, user friendliness, effectiveness of the courseware, and system response time / performance. Consultant Hu, her colleagues, and Consultant Luh were the main users of the system as the teachers/ authors. Their feedback about the system was in general favorable, and had some very valuable insights that can improve the overall effectiveness of the MMAS. The feedbacks will be further discussed in the next chapter.

The second stage of the field trial, student use, was not conducted before the end of the Project due to the delay in teachers' evaluation. (Negotiation is, however, on-going with the school and other alternatives. There is a high chance, which we certainly look forward to, to conduct a student use test in the fall semester.)

2.7 Deliverables

Early releases of the MMAS system were delivered to the Project Officer at the Education Department in July and in December 1986. The latest delivery consisted of all software modules, software drivers, all source code, the Chinese database that contains 1,000 characters, and the demo courseware.

3. Results and Evaluations

The obvious result of the Phase-II project is the completion of a working courseware system that demonstrated the capability to do instructions in multiple languages and in both the voice and the screen form. The less obvious results are, however, the intangibles, i.e. the knowledge and experience we gained during the project. These knowledge/ experience were from struggling and overcoming the many technical obstacles encountered while developing the MMAS system. We shall report some of them in the next section, Section 3.1.

Like any project at its completion, this project should also be asked the same question: How good is it ? Our answer to this question is two fold: it's definitely a success as a research project, but it needs more enhancements before becoming a successful commercial product. We shall further discuss the evaluation issue in Section 3.2, and the enhancements in Section 3.3.

3.1 Obstacles Encountered and Overcome

During the two year project, we encountered a number of technical obstacles. To overcome them in order to accomplish the project goal, we either found brilliant solutions, devised work-around, or even changed project directions. Some of those experiences are recorded here:

3.1.1 Problems with Commercial Software Packages It was decided in the early design of the MMAS that the system should be built upon as many as possible commercially available software packages. The reason was that we believed, as it is still true now, that it is cheaper to buy and integrate existing software packages than to develop them anew. Several software packages were named in the Phase-II Proposal as the foundation of our development: VDI, GKS, VDM, and the voice library. Yet, among them only the VDI and the voice library made into the final MMAS system.

The GKS was removed from the project because its slowness and in-compatibility with other packages. The GKS library provides a higher level (easier to program) interface for generating graphics and alpha-numeric text. We found, however, that the GKS is extremely slow in drawing a Chinese (or Japanese) character on the screen. Our experiments showed it took several minutes to draw a single one. The reason can be attributed to its simulating floating point arithmetics on the IBM-PC which does not have a floating point processor. We decided that speed was unacceptable and

used a feature in the VDI library to draw Chinese characters, which takes only several seconds.

Then we discovered that the two packages, VDI and GKS, could not co-exist in the same software module, as our design called for. The in-compatibility prompted our decision to drop the GKS completely from the MMAS. The cost of this decision was about 5 to 10% increase on the programming effort, since we have to use the lower level (less easy to program) interfaces provided in the VDI.

The removal of the VDM from the MMAS was mainly based on the consideration of courseware storage. The VDM was supposed to be an industrial standard to store graphic displays. Presumably more graphic displays will be available to the software that uses the standard. We found, however, that the standard was not much followed in the industry, thus following that standard ourselves would not give our system more access to other graphics tokens. We realized further that the object-oriented storage format, which we devised specifically for the MMAS, can save significantly on the disk storage space for the courseware. Consequently, we removed the VDM from the MMAS architecture and replaced with our custom-designed format.

We are pleased we made the decisions on these issues. If we did not change the directions early in the project, as we did, the project is probably still negotiating with the vendors for a faster version or struggling with the disk space. The advice for other projects should be clear: Incorporate as many commercial software packages as PRACTICALLY possible. Use others or write your own if any package does not fit your need.

3.1.2 Need for Compiler The Compiler is a module used to translate the courseware as specified by the Organizer to a machine readable form to be understood by the Rehearser. It was not in the original design, but was later recognized as essential. The advantages of the Compiler are:

1. It forms a symbol table for all the names about the screens, the voice messages, and the sections of the courseware. This eliminates the need for the Rehearser to collect the information as it starts the course. The speed of the Rehearser, and the delivery of a courseware, is significantly improved. The improvement is more obvious as the size of the courseware increases.

2. The Compiler also replaces all the names and the branches with machine readable and more efficient

addresses. That results in more compact courseware and in turn contributed to the solution of the next issue: storage requirement.

3.1.3 Courseware Storage Requirement The courseware storage requirement was discussed in the Phase-II Proposal (page 3-3) as a major obstacle for graphics oriented CAI applications. The storage requirement for a single screen was estimated (page 2-2, Phase-II Proposal) to take several thousands to hundreds of thousands of bytes, depending on the storage format to use. With that range of storage requirements, the popular low priced storage media: floppy disks, can store only minutes of courseware, and becomes commercially undesirable.

In the MMAS, the required storage for the screen information was reduced by one magnitude. This was achieved via two approaches: the object-oriented screen representation uses fewer bytes than standard graphics representations like VDM; and the MMAS Compiler processes author's specification to generate compact courseware. Each screen in the MMAS takes from one to five hundred bytes of storage, and a courseware stored on a floppy disk can last for about 2 hours,* sufficient for most applications.

3.2 How Good Is It

This Section evaluates the MMAS from two aspects: as a research experiment and as a commercial product. To evaluate the system, we shall re-visit those Phase-II Technical Objectives as stated in the Phase-II Proposal and excerpted in Chapter 1. First we address several research-oriented issues, i.e. original ideas not accomplished by other projects.

3.2.1 Provide Graphics, Text, and Voice Capabilities This objective was achieved satisfactorily. The courseware author can create courseware with color graphics, alpha-

---

* The two hour per floppy disk applies only to courseware without voice. Although the storage for the screen information was significantly reduced, that for the voice messages still take large storage space, approximately 500 bytes for each second of voice message. Consequently, making a floppy disk based courseware a reality requires more improvements in the voice message storage size.

numeric text in several fonts and sizes, and with recorded voice messages.

3.2.2 Capable of preparing and presenting in multiple languages This objective was achieved satisfactorily. The author can prepare and present courseware in Chinese, Japanese, and English, intermixed with each other. The MMAS was designed in such a modular manner that enhancing Japanese capability requires only to add the database and the database access routines. The task took only several man-months.

There were also a number of objectives that are not unique from the research aspect of view, but are nevertheless important to become a successful commercial product:

3.2.3 Friendly to the author, yet powerful enough The objective was to enable the teacher with little knowledge about computers to create the courseware using a much shorter period of time than on other existing systems. From the feedback we received from the field trial, the MMAS seems to have achieved about one half of this objective. It was considered to be general, flexible, and powerful, yet sometimes can be un-friendly to first time users. For an author who is familiar with the MMAS, she (he) can produce a courseware that meets the need in a short time period. For someone new to the system and has little or no exposure to computers, however, he (she) often has a difficult time to use the system.

During the field trial, we attempted to alleviate this un-friendliness by enhancing documentation. The result was still not satisfactory. We now conclude that it is due to too much computer knowledge required in order to effectively use the system. Some enhancements on this issue will be discussed in the Section of Enhancements.

3.2.4 Affordable The objective was to allow institutions that could not afford the high cost of computers and special peripherals for the CAI applications to have the luxury of not only educating their students, but also providing their teachers a means to create their own courseware on line. This objective has been achieved satisfactorily. THe MMAS does provide the teachers with the capability to create courseware on the PC computers. But the affordability of the system was mainly achieved by the faster than expected price dropping of personal computers. For the last three years, the price of personal computers has been dropping twenty percent or more every year. We simply picked the right hardware, personal computers, to develop the MMAS, and we benefited very much from the trend.

3.2.5 Portable To Other Delivering Systems This objective was to develop the MMAS delivering system on standard interfaces and in a modular design, so that porting to other non-IBM-PC hardware can be simplified. The ultimate goal is to use the MMAS, and the courseware developed for it, in a larger market than the IBM-PC one. During the Phase-II project, the MMAS was tested on a number of so-called PC-compatible machines, including AT&T PC, Leading Edge, and several models from the Far East. Although we tried only four to five models, the MMAS ran on all tried. It also ran on mono-graphics monitors/ adaptors, such as Hercules. From what we tested, the MMAS seems to be able to address the complete PC and PC-compatible market, itself a sufficient large one to grow.

Another smaller yet valuable market is the Apple computers, which are very in-compatible with the PC lines. Porting the MMAS to the Apples requires little or no changes if the following three interfaces exist on the Apples:

1. A C-language compiler that provides input/output interface similar to the IBM-PC Lattice C compiler,

2. A voice device and its interface, and

3. A VDI library to provide graphics interfaces,

For the Apples, there are several good C compilers that provide standard DOS interfaces. This means no change on the language interface in order to port the MMAS. There also exist several voice devices for the Apples with different interface with the Dialog/I devices. This means some changes to the MMAS voice library to port the MMAS. So as soon as the VDI library appears for the Apples, the MMAS system can be ported to the Apples in a matter of weeks. On the other hand, converting the voice courseware from the IBM-PC to the Apples is itself an interesting research subject, and warrants more study. The conversion would be valuable only when there exist large quantity of courseware for one kind of personal computers. And we are far away from that stage yet.

3.2.6 Other Objectives Other objectives that were evaluated in the field trial include the functional correctness and response time / performance. Both objectives were achieved satisfactorily. Several problems (bugs) were uncovered and fixed during the field trial. Nothing serious. The response time of the MMAS, partially because we paid special attention during the design stage, was considered quite good.

3.3 Enhancements Envisioned

This section presents enhancements that we feel necessary in order to make the MMAS a better and a more successful commercial product.

3.3.1 Friendlier Author Interface Several enhancements were felt to be desirable to make the MMAS more friendlier. They are about the high level author's interface and the keyboard interface. The first enhancement is to provide, besides the current capabilities, simple and fixed format exercises as are provided by several existing courseware packages. An example is the selection exercise: after the author selects the menu, the MMAS prompts the author to enter a Chinese (or Japanese) keyword, followed by three or four English explanations. Then the system prompts the author for the right answer, and allows the author to enter any text to display to the student who enters a wrong selection. Thus the unsophisticated teachers/authors can create simple yet useful courseware without knowing any of the sophisticated features as provided in the MMAS.

Another enhancement is for the system to recognize all input keys, so that the MMAS can process and respond intelligently. Current MMAS sometimes does not respond to other keys if it is expecting a class of input keys (e.g. function keys F1 to F10), and leaves an impression of less responsive. This enhancement requires changes to the VDI keyboard interface routines.

3.3.2 VDI Resolution Improvement An enhancement that should increase significantly the applications of the MMAS is to display more Chinese characters on the screen. Current MMAS is limited to display six rows and six columns (total 36 characters) of Chinese characters, compared with twenty-four rows of alpha- numeric characters (total almost 2,000 characters), on a normal PC screen. Teaching English-speaking students about Chinese, this is not much a problem, since most of the explanation text on the screen would be in English. This however severely limits the ability to compose courseware for teaching Chinese-speaking students about English. In the latter case, the majority text used on the screen would be in Chinese and six rows of Chinese can not convey adequate information.

The six row limitation was partly due to the resolution of the PC monitor (which has 200 times 320 pixels), partly due to the more refined patterns we used for the Chinese characters (each Chinese character is represented as a 24 by 24 matrix).

Other than using a screen with better resolution (and at a higher cost), more Chinese characters can be displayed on the screen by reducing the resolution of each character, i.e. from a 24-by-24 matrix per character to 16-by-16. Characters displayed with reduced resolution are less pretty than those with high resolution, yet are sufficient to distinct with each other. They are actually used in some Chinese word processing packages. The major work to achieve this is to acquire a Chinese database that has character representations of 16-by-16 matrix. If this can be done, the total number of Chinese characters on a screen can reach 100, sufficient for most CAI courses.

Note that the Japanese database is already in 16 by 16 representation. There is no such a restriction in constructing courseware to teach Japanese-speaking students about English.

3.3.3 More Courseware The courseware to the MMAS system is like the cassettes to the cassette recorder. Just like the cassette recorder, the MMAS is self-sufficient. It allows the creation (or the recording in the recorder case) of the courseware, and can play back the courseware. However, the MMAS would be much attractive as a commercial product if there exist a large number of courseware (or for the recorder case, pre-recorded music cassettes) for the perspective buyers to choose from. Getting more courseware written for the MMAS system will benefit tremendously toward commercializing the MMAS.

4. Potential Applications

The MMAS system is a general purpose courseware authoring and delivering system. Its screen oriented presentation, text and graphics capabilities, logic capability to control course sequences depending on student inputs, and the capability to efficiently use storage media, all make it suitable to do presentation, exercise, and drill for various classes.

The special features it possesses, nevertheless, make the MMAS most effective in foreign language instructions. These features include the capabilities to accept input and display output for Chinese, Japanese, as well as English, and the capabilities to voice pre-recorded voice messages.

Although the MMAS does not answer the students' questions intelligently, thus cannot completely substitute the language instructor's role (it was never designed to do so in the first place), it can certainly save much of the instructor's work. As was pointed out by Consultant Hu in the field trial: in between teacher's lectures, students can be assigned to take the courseware from the MMAS. Much of the teacher's current laborious responsibilities, such as providing the writing and speaking drills and measuring the progress of the students, can be accomplished by the MMAS system. The net result would be a more productive foreign language education.

MMAS Courseware № 001

INTRODUCTION: The purpose of this exercise is to practice "Sound Discrimination". Please distinguish the following words by typing in the Romanizations in pinyin and mark the tones (hit return key after each entry).

Example: From the speaker you will hear ....
for what you have heard, you will key-in the following:
fu4 jin4 **hu2 xing2**

1.	_____	_____
2.	_____	_____
3.	_____	_____
4.	_____	_____
5.	_____	_____
6.	_____	_____
7.	_____	_____
8	_____	_____
9.	_____	_____
10.	_____	_____
11.	_____	_____
12.	_____	_____
13.	_____	_____
14	_____	_____
15.	_____	_____
16.	_____	_____
17.	_____	_____
18.	_____	_____
19.	_____	_____
20.	_____	_____
21.	_____	_____
22.	_____	_____
23.	_____	_____
24.	_____	_____
25.	_____	_____

Answers:

1. huang3 hua4	谎话	huan2 gua1	还瓜
2. hui4 xie2	会写	fei4 tie3	废铁
3. gua1 fen1	瓜分	gua1 feng1	刮风
4. nao3 nu4	恼怒	lao2 lu4	碌流
5. shui3 niu2	水牛	shun4 liu2	顺流势
6. zhi1 shi4	知识	zi1 shi4	姿势奇
7. jing1 ji4	经济	jing1 qi2	惊奇
8. ji1 hui4	机会	jing4 wei4	敬畏
9. zi1 ben2	资本	zhi2 pen1	直喷
10. shi1 ren2	诗人	si3 shen2	死神
11. zheng4 zhi4	政治	zheng1 zhi2	争执
12. sheng1 ci2	生词	sheng3 zi4	省字
13. chi4 zi4	赤字	ci2 zhi2	辞职
14. shan1 shui2	山水	san1 sui4	三岁
15. si4 shi2	四十	si1 shi4	私事
16. he1 shui2	喝水	hei1 zui3	黑嘴
17. shu1 jia4	书架	zu1 ya1	租押
18. xi1 wang4	希望	xin1 huang1	心慌
19. zhi1zhu1	蜘蛛	zhu4 zhi3	住址
20. qiu2 qing2	求情	jiu4 ming4	救命
21. xiao4 che1	校车	jiao2 she2	嚼舌
22. zhi1 chi2	支持	shi1 ze2	责
23. zeng1zhi2	增值	sheng2 zhi3	识
24. chi3 cun4	尺寸	zhu2 sun3	竹笋
25. zhang1 zui3	张嘴	cang1 cui4	翠

Report:

Correct answer _____

Incorrect answer. _____

Score. _____

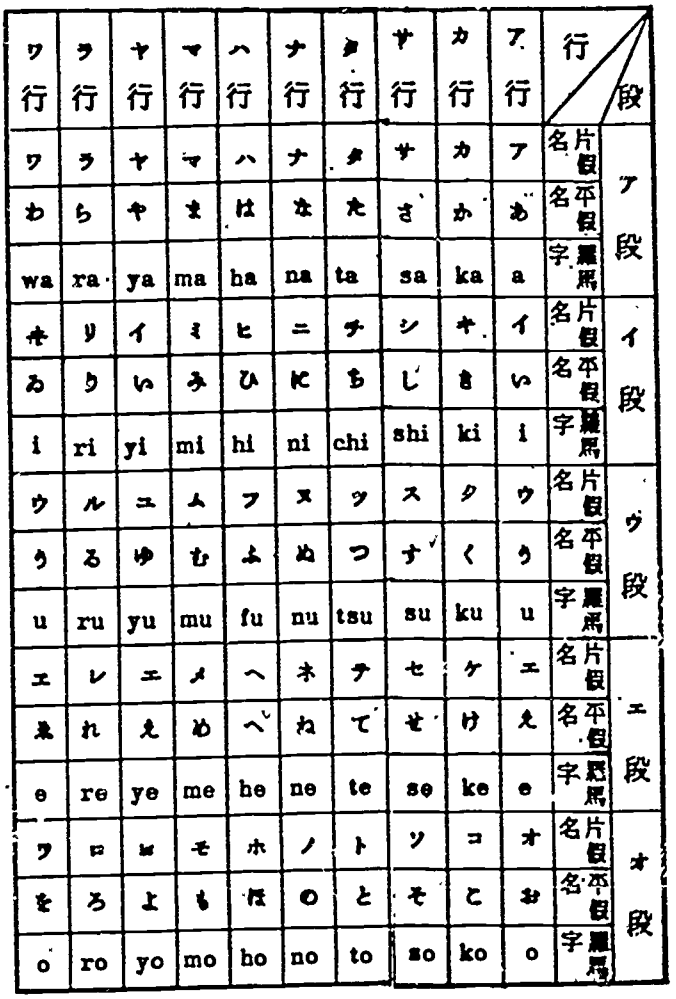
Analysis of Incorrect Answers.

_____

_____

_____

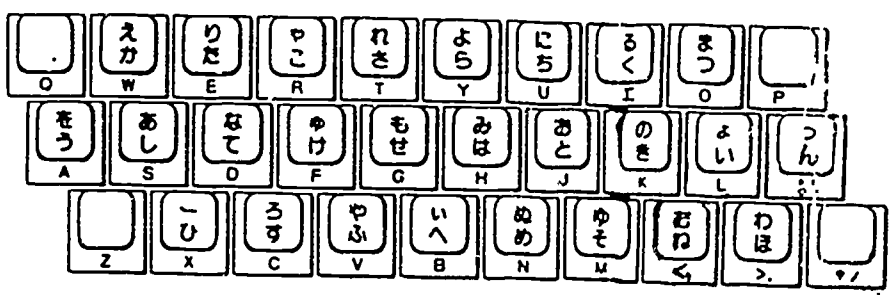
Appendix B. Japanese Phonetic Symbols and Keyboard Layout



ワ行	ラ行	ヤ行	マ行	ハ行	ナ行	タ行	サ行	カ行	ア行	行 / 段	
ワ	ラ	ヤ	マ	ハ	ナ	タ	サ	カ	ア	名 片假	ア段
わ	ら	や	ま	は	な	た	さ	か	あ	名 平假	
wa	ra	ya	ma	ha	na	ta	sa	ka	a	字 羅馬	
ヰ	リ	イ	ミ	ヒ	ニ	チ	シ	キ	イ	名 片假	イ段
ゐ	り	い	み	ひ	に	ち	し	き	い	名 平假	
i	ri	yi	mi	hi	ni	chi	shi	ki	i	字 羅馬	
ウ	ル	ユ	ム	フ	ヌ	ッ	ス	ク	ウ	名 片假	ウ段
う	る	ゆ	む	ふ	ぬ	つ	す	く	う	名 平假	
u	ru	yu	mu	fu	nu	tsu	su	ku	u	字 羅馬	
エ	レ	エ	メ	ヘ	ネ	テ	セ	ケ	エ	名 片假	エ段
衣	れ	え	め	へ	ね	て	せ	け	え	名 平假	
e	re	ye	me	he	ne	te	se	ke	e	字 羅馬	
ヲ	ロ	ヨ	モ	ホ	ノ	ト	ソ	コ	オ	名 片假	オ段
を	ろ	よ	も	ほ	の	と	そ	こ	お	名 平假	
o	ro	yo	mo	ho	no	to	so	ko	o	字 羅馬	

五十音圖

Japanese Phonetic Symbols



Fujitsu Keyboard Layout

Appendix C. MMAS Version 1.0 User's Manual

MMAS VERSION 1.0
USER'S MANUAL

February 1, 1987

1. INTRODUCTION

MMAS is a multi-language, multi-media authoring and delivering system for courseware. The primary goal of MMAS is to improve the productivity for creating and editing courseware. The system provides non-programmer authors with an integrated tool set to write courseware for foreign language teaching using multiple media, such as text, graphics and voice. The written courseware can be delivered on inexpensive personal computers or microcomputers available or affordable at homes or teaching institutions. MMAS isolates the device-dependent component from courseware so that courseware can run on many different machines without changes. This feature improves courseware portability significantly. The system has a menu-driven interface which allows non-programmer authors or teachers to write courseware for individualized learning.

The MMAS version 1.0 runs on IBM PC, PC/XT or PC/AT with DOS version 2.X and 3.X with at least 256K memory. The graphics boards, printers and peripherals that can be potentially used in version 1.0 are listed in Appendix 1. To configure a MMAS system, the graphics board(s), printer(s) and other peripherals used in the target PC need to be specified. In version 1.0, a Chinese database is included. This makes it easy to write courseware for teaching English-speaking students Chinese and Chinese-speaking students English. It has the capabilities of displaying English text, Chineses characters, graphics objects, and generating the voice recorded by the courseware authors. MMAS also provide tools for organizing the units in a course, sequencing the instruction flow, making decision based in the inputs, and looping through the teaching session.

The version 1.0 of MMAS will be upgraded and upwardly compatible with new features in the future, such as running on more types of machines and operating systems like UNIX, adding databases for various languages like Japanese and Arab, enhancing the media dimension like IV video.

The manual is organized into the following sections:

- Description of the Package

- Installation
- Getting Started
- System Description
- Organizing Courses (COMPOSE)
- Editing Graphics (GED)
- Editing Voice (VED)
- Compiling Courses (TRANSLATE)
- Rehearsing Courses (REHEARSE)
- Making Courseware (EXTRACT)
- Delivering Courseware (DELIVER)

2. DESCRIPTION OF THE PACKAGE

In this MMAS version 1.0 distribution, there are three floppy disks: two for the authoring system and one for the student (delivering) system. One floppy disk for tutorial and sample courseware will be available soon. The authoring system contains four types of files: (1) MMAS executable files: compose.exe, ged.exe, ved.exe, translate.exe, rehearse.exe, and extract.exe; (2) foreign language database files: chinese.fnt and chinese.idx; (3) device driver files: driver.exe, gssvdi.sys, voice.sys, and disp.sys (graphics device dependent); and (4) utility files: mmas.pat, mode.com etc. The student system basically contains the same files except that only one MMAS executable deliver.exe is included.

3. INSTALLATION

In installation, the following five steps need to be done: (1) copy the files on distribution disks to a proper directory, (2) modify autoexec.bat, (3) modify config.sys, (4) modify mmas.pat and (5) install drivers and database files in the proper places. We now discuss the details for the case of the authoring system. The procedures for the student system is basically the same unless stated otherwise.

3.1 Copying To A Directory

We assume the target machine being installed has a hard disk. Other configurations will be added later when the size of the MMAS software gets stable. In the target machine, make a directory "mmas" in disk c (hard disk) for the MMAS system by typing "mkdir mmas". Then change the working directory to mmas by typing "cd mmas". Now one can put one proper distribution disk in floppy disk drive a. And copy all the files from the distribution disk(s) to this directory by typing "copy a:*.* .". After this is done, copy another distribution disk in the same way for the authoring system.

3.2 Modify autoexec.bat

Add the following three lines in the autoexec.bat file in the c:\ directory:

```
set display=c:\mmas\disp.sys
set c:\mmas\disp.sys = con
c:\mmas\drivers
```

3.3 Modify config.sys

Add the following lines in the config.sys file in the c:\ directory:

```
device = c:\mmas\disp.sys
device = c:\mmas\gssvdi.sys
device = c:\voice.sys
```

If the target machine does not have a voice board, then the third line should be omitted.

3.4 Modify mmas.pat

Since the database files for foreign language are relatively large, the user has the option to put chinese.idx and chinese.fnt in other directory of other disk. This is especially useful when the target machine does e hard disk. If those database files are going to be put in put in a directory, say b:\chinese, then the two lines in the mmas.pat file should be modified as follows:

```
font = b:\chinese\chinese.fnt
index = b:\chinese\chinese.idx
```

3.5 Putting Device Driver and Database Files

The final step is to move all the driver file and database files to proper directories. If a voice board is in place, then copy the voice.sys file to c:\ by typing "copy voice.sys ..". If the database files will not be in c:\mmas, then the user should make sure those files should be put in the directory indicated in the mmas.pat file.

4. SYSTEM DESCRIPTION

We first discuss the authoring system. After the system is booted, the author can move to the directory mmas, and then starts to use the following tools to write courseware. There are currently six authoring tools: (1) "compose" for organizing courseware in units, (2) "ged" for editing graphics components for the course units, (3) "ved" for editing voice components for the course units, (4) "translate" for translating and linking units into courseware, (5) "rehearse" for rehearsing the courseware, and (6) "extract" for extracting all the needed files to make a floppy disk containing courseware.

A top-down design methodology is recommended as follows. The author first uses "compose" to describe the relationship of the course units and the control flow within each unit. The graphics and voice components are represented by a file name for later detailed design. The second step is to use the graphics editor "ged" to layout the graphics for the graphics files described in the units created by "compose". The third step is to use voice editor "ved" to record the voice needed in the course units. After these three steps are done, the author can run "translate" to see if the design has any errors or unspecified names. If it goes well, the courseware is made in a runnable form. Once the runnable form is obtained, the author can invoke "rehearse" to run the courseware. Finally the author can invoke "extract" to make courseware for distribution.

"rehearse" is slightly different from the student system's "deliver" tool in that it may provide some debugging capabilities for helping authors to find out the problems. Currently, such capabilities are very limited for authors and will be enhanced later on. The user in a student system only needs to know the course name, and run it using the "deliver" tool.

Currently, the courseware should be made in the working directory. Each course consists of units. The course file has the suffix .cor and the unit file has the .unt suffix. A

unit may contain voice or graphics components stored in files. The voice file is suffixed with .vos for voice file, while the graphics file is suffixed with .scr. The translated courseware has the .run suffix.

5. ORGANIZING COURSEWARE (COMPOSE)

5.1 Get Started

At the DOS command level, this tool can be invoked by typing "compose" in the working directory. It can take two types of options: one is "-u" followed by a unit name, and another is "-c" followed by a course name.

5.2 Introduction

In the MMAS authoring environment, the author uses the "COMPOSE" program to specify (1) the structure of a course, in terms of units, and (2) the function of individual units in the course. "COMPOSE" provides a friendly menu-driven command interface (using function keys F1 - F8) to facilitate these tasks. The set of commands for task (1) collectively is called the Course Structure Composer (CSC). The other set of commands for task (2) is called the Unit Flow Composer (UFC). The Course Structure Composer is most useful in managing the modularity required by the advanced courseware development. We shall elaborate on this subject after we discuss other basic capabilities provided by "COMPOSE". Let's start with the simplest course structure - a course consists of only one unit. In this case the course's behavior is completely determined by the unit. We shall examine in Section 5.3 the basics of a unit. These include a set of instructions that allow the author's intent to be captured in a courseware program. The reader can find a sample of detailed steps taking place in creating a single-unit course in Ref[ ]. Then in Section 5.4 the view of a course will be extended to cover multiple units through the use of the Course Structure Composer. Section 5.5 is a summary of the "COMPOSE" menu.

5.3 Basics of Courseware Programming

The sequence of operations within a unit is determined by a courseware program. There are 3 major categories of operations in a courseware program: (1) inputs/outputs, i.e. the interactions with the student; (2) data, which may be received from inputs or generated internally by the program; and (3) decisions, implementing the intelligence of choosing different actions for different situations. At the detail level these operations appear in the form of instructions,

each being an abbreviated "op-code" followed possibly by parameters. A good courseware program is a sequence of instructions properly organized to deliver the proceedings of a course. The execution of this program presents the course to the student.

The MMAS delivering environment supports 2 forms of inputs and 3 forms of outputs. One input form exercised by a text input instruction causes information to be stored in one of 4 types of data variables, namely integer, real number, Chinese string and generalized English string (allowing alphabets, numerals and printable characters on the keyboard). The other input form exercised by the function-key input instruction puts a number between 1 and 10 in a data variable of type "key", depending on which of the F1-F10 keys is being pressed. The 3 output forms are text, graphics and voice. A text output instruction displays a Chinese string, a generalized English string, or the content of a data variable. This display is shown in a special area at the bottom of the screen, right next to the previous text in the same area. This line is used for transitory dialog, and can be cleared by the "Clear Line" instruction. A graphics output instruction displays the graphics stored in a screen file. This file was created by the author using the "GED" program. The full screen is available for graphics. The graphics caused by a screen file can also be erased by the "Erase" instruction. The entire screen can be cleared by the "Clear Screen" instruction. The voice output instruction plays a recorded message on the speaker. The message was stored in a voice file by the author using the "VED" program.

As mentioned above, data types in a courseware program are integer, real number, Chinese string, generalized English string and key. They have a type indicator, namely I, R, C, S and K, respectively. Data variables are identified by a unique name 3 to 8 characters long. The first character is the type indicator, the second is underscore "_" and the rest are alphabets or numerals. When a data variable appears in an instruction, it is either receiving a value or being used as a known datum. Logically it should have received a value before this value can be used. Like the input instructions described above, the "Assign" instruction lets a data variable receive a value. The "Assign" instruction includes an assignment specification in the form of "variable=expression". The data variable at the left hand side receives a value equal to the computed result of the expression at the right hand side. An expression can be a number, a data variable, or well formed by data variables and operators such as addition(-), substraction(-), multiplication(*) and division(/). Parentheses can be used

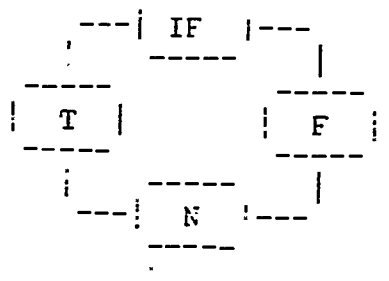
like in any mathematical equations.

Example:

```
* update sum and average over a number of scores:
* increment the number of scores
ASSIGN  I_NO=I_NO-1
* add the new score to sum
ASSIGN  R_SUM=R_SUM-R_SCORE
* compute average
ASSIGN  R_AVE=R_SUM/I_NO
* done!
```

Decisions in a courseware program are achieved by using the IF instruction. This instruction has a condition specification. A simple condition is a comparison between two data variables, being specified by inserting a comparison operator between their names. Comparison operators are "equal to" (=), "not equal to" (!=), "greater than" (>), "less than" (<), "greater than or equal to" (>=), and "less than or equal to" (<=). When a simple condition is evaluated at the delivering environment, it yields a true/false result depending on whether the comparison relationship is satisfied. Parentheses are added around simple conditions for clarity, when they are joined together to form a complex condition. The logical-and (&) and logical-or (|) operators are used in this case. These operators combine the true/false results of simple conditions in the logical manner.

The order of execution beyond the IF instruction depends on the result of evaluation. The dependency is shown below:

line no.		
n:	IF instruction	
n+1:	"true" instruction T	
n+2:	"false" instruction F	
n+3:	"next" instruction N	

Each of these instructions occupies one line in the courseware program. If a true/false branch does not require any action, a comment line is needed to hold the space.

Example:

```
IF      K_SELECT = 1                          ---------------
ASSIGN  I_GOOD = I_GOOD - 1                      <-- True ---!
.*      no-op if false                           <-- False --!
                                                 <-- Next ---
```

Note that the flow indication at far right is displayed by the Unit Flow Composer. This is useful to remind the author about this unique multi-instruction structure.

Since the above structure allows only one instruction for each true/false branch, the CALL instruction can be used there for re-directing the execution to elsewhere in the courseware program. The CALL instruction has a routine name as its parameter, which should also appear in a unique ROUTINE instruction. The execution transfers to the instructions following the ROUTINE instruction. It will then come back to resume after the CALL instruction as soon as a RETURN instruction is met. This CALL-ROUTINE-RETURN capability is very useful in authoring sizable courseware programs. Certain sequences of operations repeatedly used are the best candidates to be written as routines. Program readability can be enhanced by using them.

Similar to CALL can ROUTINE, the JUMP can LABEL instructions allow transfer of execution sequence. But they are normally used without the intention of coming back. Two other instructions exist: the EXIT instruction causes the courseware program to terminate; the PAUSE instruction causes the computer to sit idle for a number of seconds as specified by its parameter. Both instructions are essential for courseware programming.

5.4 Hierarchical Design of a Courseware

When a courseware is complex, it becomes cumbersome to deal with a huge unit file. It is especially inconvenient if frequent revisions are expected or different courses are to share some part of the material. We recommend the user to take modular and structural design into consideration. A course can be designed as a collection of units. Each unit can be a high level module in the course, or a low level sharable utility routine, such as data collection and reporting. The relationship among units is characterized by the CALL-and-RETURN type of transfers. We can view the structure of a courseware as a tree: the root of the tree is the unit where execution starts, the units it invokes are its children. These units can also invoke other units (their children), and so on. The CSC is designed to capture this relationship. It offers to the user the capabilities to construct this structure tree, to trim this tree, and to

move a branch of the tree from one place to another. It also allows the structure tree of one course to become a branch in the tree of another course.

The COMPOSE program writes the structure tree of a course into a file named <coursename>.cor. When the TRANSLATE program is executed, given the course name, the file will be examined. The names of its units will be extracted and proper unit files, <unitname>.unt (as composed using the UFC), will be compiled. The final translation result will be a file named <coursename>.run ready for REHEARSE and DELIVER. In case of a single-unit course, it is required to create a structure "tree" consisting only of the unit (the "root"). A This can be done by just one command: "add" in the eighth menu described below.

5.3 Menus

There are nine menus in COMPOSE. The first menu is the control menu for selecting a composer to work on and saving the result in files. The second menu is the main menu of the UFC, which is for locating, insertion and deletion of the statements in a unit. The UFC displays 20 consecutive statements on a screen. The blinking cursor on the screen indicates the position on which the next operation will be applied. The third, fourth and fifth menus elaborate on the insertion mode, and they are for writing statements in a unit. The sixth menu is for selecting the data form in an output statement. Similarly, the seventh menu is for selecting the input data form. The eighth menu is the entry of the CSC. The eighth and ninth menus are for composing the hierarchical structure of a course. The CSC displays the structure graphically on the screen. The unit being tagged with asterisk indicates the current cursor position.

5.3.1 The First Menu The first menu has eight entries. The first entry "end" exits from COMPOSE. The second entry "comp cor" enters the CSC (the eighth menu). The third entry "load cor" loads a course from a .cor file and enters the CSC. The fourth entry "save cor" saves the working course structure to a .cor file. The fifth entry "load unt" loads a unit from a .unt file and enters the UFC (the second menu). The sixth entry "save unt" saves the working unit into a .unt file. The seventh entry "comp unt" enters the UFC. The last entry "help" is reserved for the on-line help.

5.3.2 The Second Menu The second menu has eight entries. The first entry "end" exits from the UFC and moves to the first menu. The second entry "insert" enters the statement insertion mode, which would lead to the third menu. The third entry "delete" allows the user to delete one or more

statements. The fourth entry "recall" inserts the statements removed by the last delete in front of the current cursor. The fifth entry "prev st" moves the cursor to the previous statement. The sixth entry "next st" moves the cursor to the next statement. The seventh entry "prev pg" moves the cursor to the previous page. The eighth entry "next pg" moves the cursor to the next page.

5.5.3 The Third Menu There are seven entries in the third menu. The first entry "end" terminates the insertion mode and moves to the second menu. The second entry "ASSIGN" allows the user to enter an assignment statement. The third entry "IN" enters an input statement in the current cursor position, and the seventh menu will show. The fourth entry "OUT" enters an output statement in the unit, and the sixth menu will show. The fifth entry "IF" allows the user to enter a condition. The 'then' consequence should appear one line below this condition statement and the 'else' consequence at the line further below. As states in Section 5.3, a null statement is needed to fill the line if either "then" or "else" has no consequence. The sixth entry "." enters a null statement which can be attached with a comment. The last entry "etc" moves to the fourth menu described below.

5.5.4 The Fourth Menu There are eight entries in the fourth menu. The first entry "end" behaves the same as the "end" in the third menu. The second entry "RETURN" enters a return statement for function return. The third entry "CALL" allows the user to enter a call statement for calling another functions within the same unit or another unit. The fourth entry "ROUTINE" enters the routine statement as the entry point of a function. The fifth entry "JUMP" allows the user to enter a jump statement to transfer execution to a label. The sixth entry "LABEL" enters a label statement in the current cursor position. The seventh entry "EXIT" enters an exit statement to allow exit from a course. The last entry "etc" moves to the fifth menu.

5.5.5 The Fifth Menu The fifth menu has six entries. The first entry "end" is the same as in the third menu. The second entry "PAUSE" enters a pause statement to indicate the pause time. The third entry "CLR SCR" enters a clear screen statement in the current cursor position. The fourth entry "CLR LNE" enters a clear line statement in the unit. The fifth statement "ERASE" enter an erase statement that can erase the objects represented by a .scr file from the screen. The last entry "etc" moves to the third screen.

5.5.6 The Sixth Menu The sixth menu appears while an output statement is entered. There are six entries in this menu. The first entry "CHI STR" indicates the intention to output a Chinese character string. Once it is chosen, the user is asked to enter the pin-yin code for Chinese characters and select the right one if more than one homonym exists in the library. The second entry "ENG STR" allows the user to enter a generalized English text string for output. The third entry "VARIABLE" allows the user to enter a variable name. The value of the variable will be output. The fourth entry "VOICE" allows the user to indicate a voice output using a .vos file. The fifth entry "GRF" indicates a graphics output using a .scr file. The sixth entry "PAR GRF" allows the user to enter an output statement with parameters to be overlaid with the graphics in a .scr file.

5.5.7 The Seventh Menu The seventh menu appears while an input statement is entered. The result of the input is to be stored in a variable. There are five entries in this menu. The first entry "INTEGER" specifies an integer variable. The second entry "REAL" specifies a real variable. The third entry "CHI STR" specifies a Chinese character string as the input. The fourth entry "ENG STR" allows the user to enter an English text string variable. The fifth entry "FUN KEY" enters an input statement with function key as input.

5.5.8 The Eighth Menu This is the entry for the CSC. The eighth menu has eight entries. The first entry "end" exits from the CSC and moves to the first menu. The second entry "zoom in" enters the UFC and moves to the second menu. COMPOSE will look for the file named <unitname>.unt, where <unitname> is the name of the unit pointed by the current cursor. This file is used to pre-load the UFC working area, unless the file is not found and the working area is unchanged. The third entry "up" moves the cursor up to the current unit's parent in the structure tree. The fourth entry "add" adds a new child to the current unit. This is also used to enter the root when the structure tree is empty. The fifth entry "left" and the sixth entry "right" moves the cursor to a sibling unit from the current unit. The seventh entry "down" moves the cursor from the current unit to its child. The last entry "etc" moves to the ninth menu.

5.5.9 The Ninth Menu The ninth menu has eight entries. The first entry "end" is the same as the "end" in th eighth menu. The second entry "search" allows the user to search for a unit by name in the CSC working area. The cursor will move to the unit, if found. The third entry "up alt" allows the cursor to move to a parent of the current unit on an

alternative path in The fourth entry "add ext" reads a given .cor file and attach the structure tree stored in the file (as descendants) to the current unit. The fifth entry "delete" deletes the current unit and its descendants from the structure tree in the working area. The sixth entry "recall" adds to the current unit new descendants that were removed by the last delete. The seventh entry "chg root" makes anew structure tree where the current unit becomes the new root. The last entry "etc" moves to the eighth menu.

6. EDITING GRAPHICS (GED)

6.1 Introduction

Graphics Editor (GED) is the graphics editor for the MMAS system. It can be used to create screens of graphics for the monitors with graphics capability. The graphics files created by GED can be used as a part of the courseware created by the MMAS system.

GED can be used to create graphics composed of a variety of objects. An object can be an English text, one foreign language text (for version 1.0, this is the Chinese text), and a graphics object. such as line, filled area, and marker. Besides creation of objects, it provides capabilities to move the objects on the screen and to delete the objects. Other attributes of an object that can be changed include its color, size, and a number of object-specific attributes.

Like many newer software packages, GED is menu driven with numerous help messages. With some understanding about the basic usages of GED, a user can explore through the package without much need for the manual.

When you need to save the screen, either for later editing or for use in the courseware, you can use the save capability to keep the whole screen in a disk file. You will be prompted for the name of the file to save. More details can be found in the menu section. The files created in the GED session are always prefixed with an extension .SCR. The sizes of the files range from from 150 bytes to about 3000 bytes each, depending on the number of objects on the screen.

6.2 Basic Concept

A set of basic concepts, if kept in mind, would benefit much in using the package.

6.2.1 Organization of Menus The menus are organized in layers. When the program is first started, the main menu is presented at the bottom of the screen. By entering program function keys (F1 to F10), the user can select the particular functions. Entering a function key often replaces the current menu on the screen with another, lower level menu. Any time when an Exit function (always F1) is entered, the current menu is replaced with a previous or a higher level menu.

6.2.2 Cursor Position An important notion is the cursor position. It indicates the attention of the system. The creation of an object takes place on the screen at the cursor position. Also, only when the cursor is positioned on an object most operations like deletion, moving, coloring, sizing, and creation can be performed. So the right procedure in using GED is to always move the cursor to the chosen position, then perform the operations desired.

To find out the current position of the cursor, enter the CURSOR function key (F2 on the keyboard) when the main menu is displayed on the bottom of the screen. The cursor position is displayed on the screen as a cross (a plus sign +). To move the cursor, use the four position keys ->, <-, etc, on the right side of the keyboard, to move the cursor to the desired position on the screen. Entering RETURN key freezes the cursor position.

If an operation is to perform to an existing object, the cursor must be positioned onto that object first. Because an object may occupy a large area of the screen and several objects may overlay with each other, one unique way is necessary to identify the object. The convention used in GED is that the cursor must be close to the left most part of the object. When more than one part of an object is the left most, the lowest part is consider the position of the object. As an example, an box shaped object would have a whole line (left vertical side) as the left most part. With this rule, the left, lower corner is the position of the object. To delete or move that object, the cursor must be moved to the left lower corner first.

6.2.3 Help Menu Whenever a menu is displayed at the bottom of the screen, enter function key PF9 or PF10 would always gives the help menu. The help menu explains the current options to the user and their effects. Feel free to use

whenever you don't know what to do.

6.3 Get Started

At DOS command level, GED can be invoked by entering either

```
ged
```

or

```
ged filename
```

where filename is the name of the graphics file that was saved in a previous edition session. An example of invoking the program is

```
ged demo
```

Note the filename should not have any extension. The following is an INCORRECT example:

```
ged demo.SCR
```

After a few seconds of starting period, the screen should be cleared and the following display, the main menu, should display at the bottom of the screen:

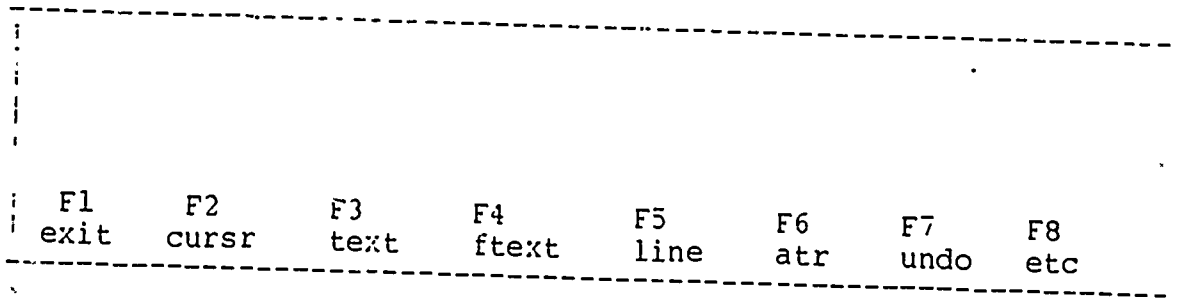


Figure 1

Each function indicates an operation to perform. Entering F9 or F10, which are not shown on the screen, would bring up the help menu.

Suppose F2 is entered, the menu should be replaced by this message:

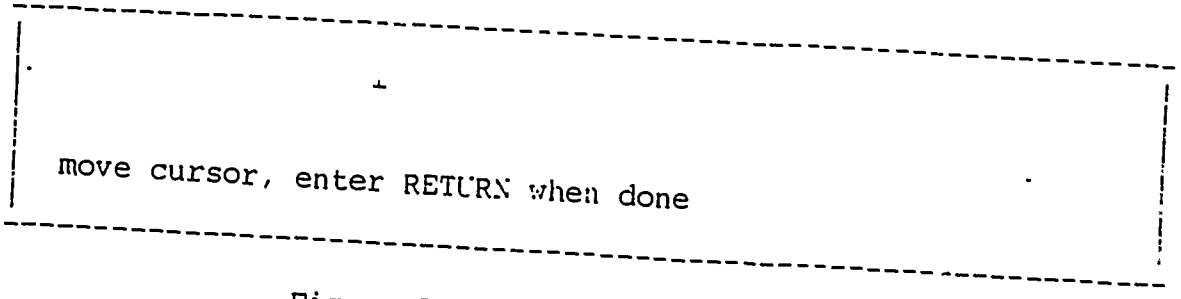


Figure 2

The cursor position is displayed by a cross on the screen. For the first time, the cursor is at the left, lower corner. Only a part of the cursor can be seen. Now one can use the cursor keys on the right portion of the keyboard to move cursor. For example, the key marked with "6" and "->" would move the cursor toward the right direction; the key marked "9" and "pgup" would move the cursor toward the right and up direction. When the desired direction is reached, one enters RETURN key to signal to the system. The main menu should return to the screen.

Now, suppose one enters the text function key (F3) to create an English phrase. The screen should read "enter text: ". Suppose one enters the word "Test", it would first show at the bottom of the screen. After the RETURN key is entered, it appears on the screen where the cursor was, and the main menu returns. The screen is shown in the following. An English text object has been created.

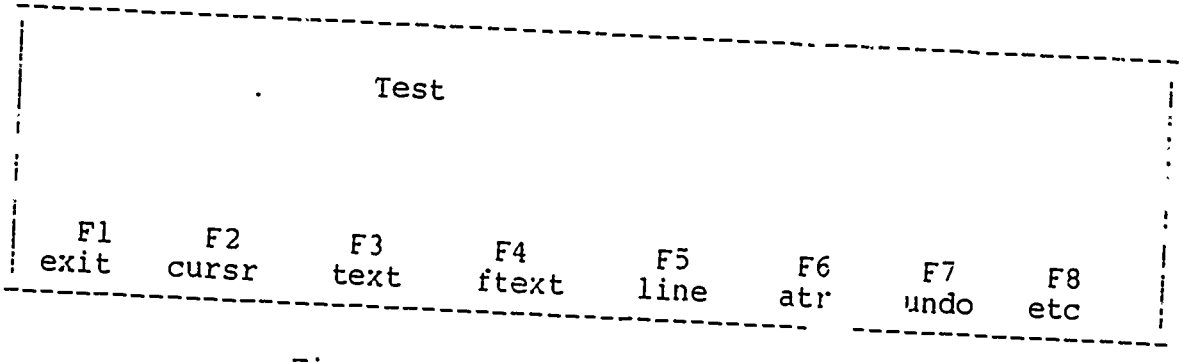


Figure 3

Now suppose one wishes to change the color of the text object just created. Entering F6 key is to bring out the attribute menu for the text just created. There is no need to move cursor because the cursor is still positioned to the

text object. In situations to modify attributes for an object that is not currently positioned by the cursor, the cursor must be moved first to position to the object. The new screen looks like:

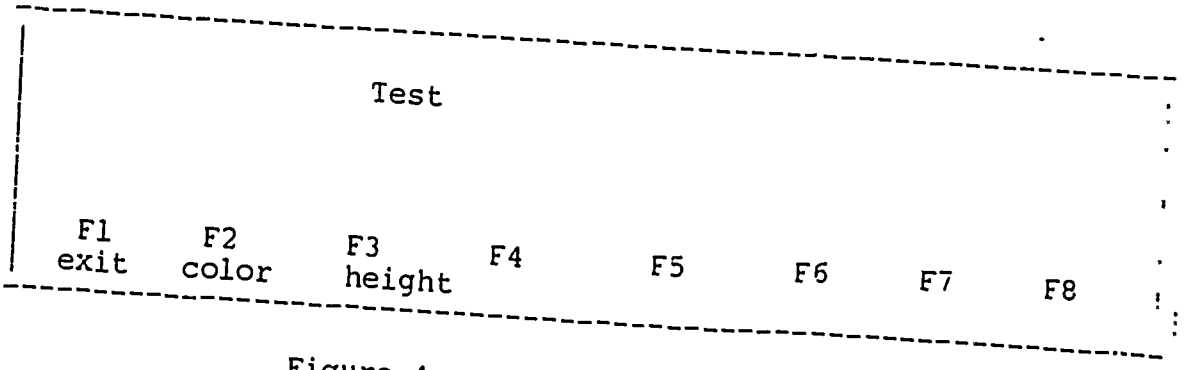


Figure 4

Entering F2 again is to bring yet a menu with available color alternatives.

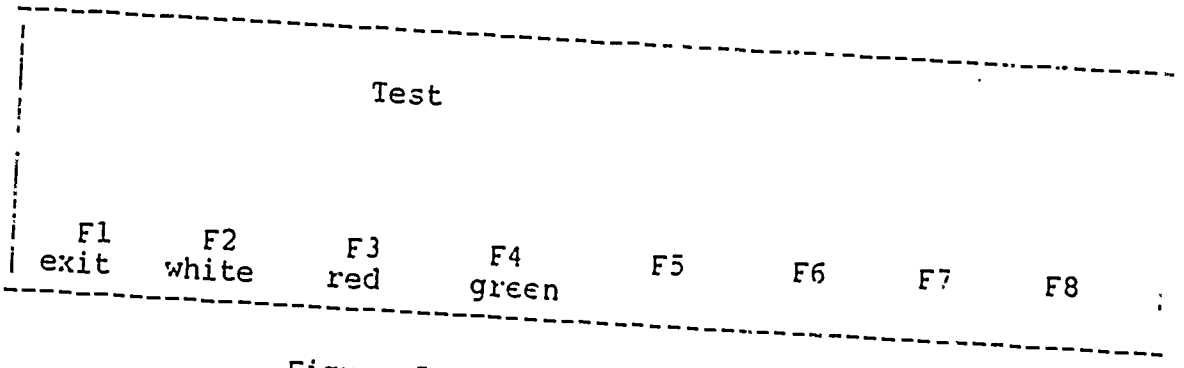


Figure 5

Entering F4 now is to change the color of the text "Test" from previously white to currently green. The menu is to return to the attribute menu, i.e. Figure 4. In case you want to go back to the main menu (Figure 3) where we started, enter F1 (Exit). This would exit from the current menu to the previous menu.

The following Figure tells the relation among the menus we discusses in this section.

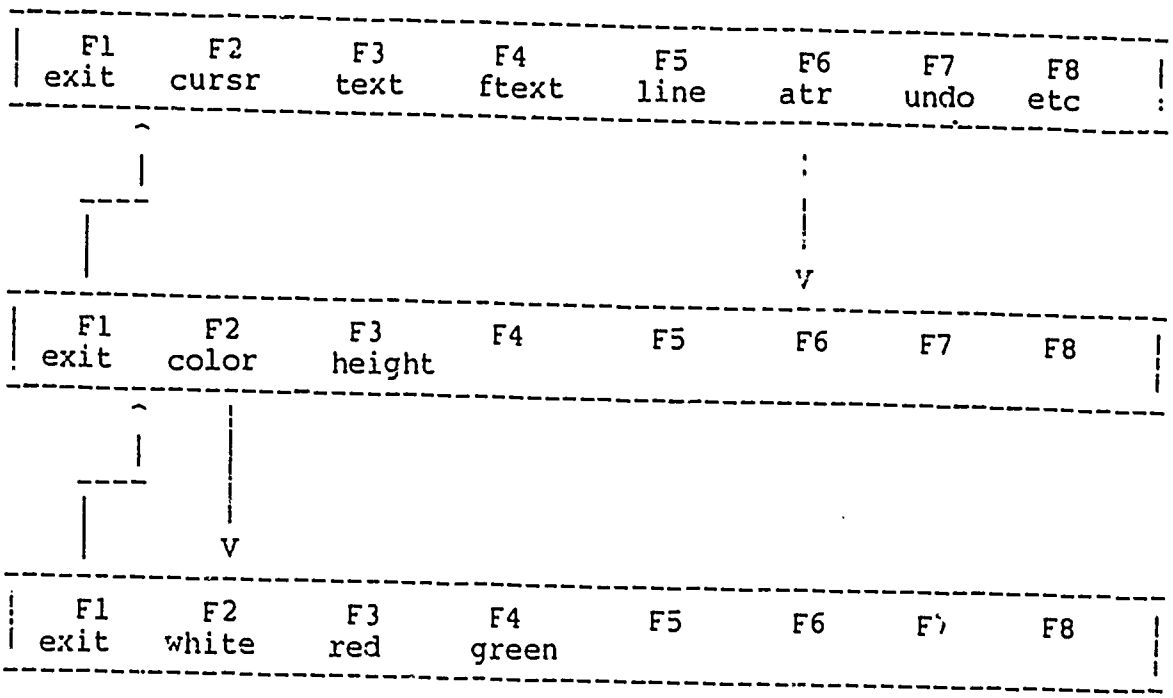


Figure 6

6.4 Menus

There are thirteen menus in GED. Each of the menus is first listed, followed by descriptions for each field.

6.4.1 Menu 1

F1 exit	F2 cursr	F3 text	F4 ftext	F5 line	F6 atr	F7 undo	F8 etc

1. exit - to exit from the program;
2. cursr - to move the cursor;
   After entering F2, use keys on the right of the keyboard <-, ->, etc. to move cursor.
   Enter return when reach the destination;
3. text - to create English text;
4. ftext - to create foreign (Chinese) text;

5. line - to create a line; When entered, a symbol s is to appear at where the cursor is. This is also the start of the line. Use the cursor movement keys on the right side of the keyboard to move the cursor to the next position. Then enter return key. Repeat the step to get the next points of the line. To EXIT, however, the cursor must be positioned back at the starting symbol s and the return key entered;

6. atr - to change the attribute for an object, which can be a text, ftext, line, area, or marker; Depending on the object, the attribute can be changed may include the color, size, pattern, etc.

7. undo - to undo the previous creation or deletion;

8. etc - to display the next menu, i.e. Menu 2.

6.4.2 Menu 2

F1 exit	F2 cursor	F3 move	F4 del	F5 draw	F6 save	F7 load	F8 etc

1. exit - to exit from the program;

2. cursor - to move the cursor;
   After entering F2, use keys on the right of the keyboard <-, ->, etc. to move cursor.
   Enter return when reach the destination;

3. move - to move an object. The cursor must be first pointed to the left, lower corner of the object;

4. del - to delete an object; The cursor must be first pointed to the left, lower corner of the object;

5. draw - to repaint the screen, or repaint only the object if the cursor is pointed to an object;

6. save - to save the screen on a disk file; You will be prompted for the name of the file. Do not give the extension of the file. The extension will always be .SCR. For example, you answer the prompt with "mountain," the file created will appear on disk as mountain.SCR.

7. load - to load the screen image from a disk file; You will be prompted for the name of the file.

8. etc - to display the next menu, i.e. Menu 3.

6.4.3 Menu 3

F1	F2	F3	F4	F5	F6	F7	F8
exit	cursr	mark	area	debug	help	---	etc

1. exit - to exit from the program;

2. cursr - to move the cursor;
   After entering F2, use keys on the right of the keyboard <-, ->, etc. to move cursor.
   Enter return when reach the destination;

3. mark - to create a marker; When entered, a symbol S is to appear at where the cursor is. This is also the start of the marker. Use the cursor movement keys on the right side of the keyboard to move the cursor to the next position. Then enter return key. Repeat the step to position the next points of the marker. To EXIT, the cursor must be positioned back at the starting symbol $ and the return key entered;

4. area - to create a filled area; When entered, a symbol S is to appear at where the cursor is. This is also the start of the marker. Use the cursor movement keys on the right side of the keyboard to move the cursor to the next position. Then enter return key. Repeat the step to position the next points of the marker. To EXIT, the cursor must be positioned back at the starting symbol $ and the return key entered;

5. debug - to enter debugging mode; This is to create large files on the disk. Not recommended to use.

6. -- - to display the help menu;

7. etc - to display the next menu, i.e. back to Menu 1 now.

6.4.4 Menu 4

F1	F2	F3	F4	F5	F6	F7	F8
exit	color	height					

This menu is an attribute menu. It is used to change the color and the height of the text object.

1. exit - to go back to Menu 1;
2. color - to change the color;
3. height - to change the height;

6.4.5 Menu 5

F1 exit	F2 color	F3 style	F4 inter	F5	F6	F7	F8

This menu is used to change attributes for the area object.

1. F1 - to exit from the menu to Menu 3;
2. F2 - to change the color;
3. F3 - to change the style;
4. F4 - to change the interior;
5. F5, F6, F7, F8 - to get the help screen;

6.4.6 Menu 6

F1 exit	F2 color	F3 type	F4 width	F5	F6	F7	F8

This menu is used to change attributes for a line object.

1. F1 - to exit from the menu to the main menu;
2. F2 - to change the color of the line;
3. F3 - to change the type of the line;
4. F4 - to change the width of the line, not provided with IBM-PC;
5. F5, F6, F7, F8 - to get the help screen;

6.4.7 Menu 7

F1 exit	F2 color	F3 type	F4	F5	F6	F7	F8

This menu is used to change attributes for a marker object.

1. F1 - to exit from the menu to Menu 3;
2. F2 - to change the color of the marker;
3. F3 - to change the type of the marker;
4. F4, F5, F6, F7, F8 - to get the help screen;

6.4.8 Menu 8

F1	F2	F3	F4	F5	F6	F7	F8
exit	white	red	green				

This menu provides selections on the color of an object.

6.4.9 Menu 9

F1	F2	F3	F4	F5	F6	F7	F8
exit	0.3"	0.5"	0.7"	1"	1.5"	2"	

This menu provides selections on the height of a text or a foreign text.

6.4.10 Menu 10

F1	F2	F3	F4	F5	F6	F7	F8
exit	solid	dash	dot	ddot	mdash	ddot	sdash

This menu provides selections on the various types of a line.

6.4.11 Menu 11

F1	F2	F3	F4	F5	F6	F7	F8
exit	dot	plus	star	O	X	diamond	

This menu provides selections on the various types of a marker.

6.4.12 Menu 12

F1	F2	F3	F4	F5	F6	F7	F8

C-21

exit	hollow	solid	pat	hatch

This menu provides selections on the various interiors of an area.

6.4.13 Menu 13

F1	F2	F3	F4	F5	F6	F7	F8
exit	narrow	med	wide	2narrow	2med	2wide	

This menu provides selections on the various styles of an area.

7. EDITING VOICE (VED)

7.1 Introduction

The author can use the voice editor to create or edit voice files for the course via the microphone and the speaker connected to the PC. The author can only work at one file at a time.

7.2 Get Started

This voice editor is menu-driven. It is invoked by typing "ved" (stands for voice editor). Then the system prompts with some questions on initialization. If non-default is chosen, then the system asks the author to enter sampling rate, encode option, and length limitation in sequence. Otherwise the default setting is chosen with 4K per second sampling rate, encoded, and 15 second length limit.

A played file should have the same set of parameters as those when it was recorded. To avoid unexpected mismatch, it is advised to use the default setting always. This is done by typing "0" for the first question.

Once a set of parameters are chosen, all the playing or recording session will use the same parameters until the author decides to exit from the voice editor.

7.3 The First Menu

After the initialization session, the first menu with six entries shows at the bottom of the screen. The menu is mainly for manipulating voice files. The author can then select the menu using the function key F1 through F6.

We now discuss each entry in this first menu.

The first entry "create" is used to create a voice data file. Once the entry is chosen, the system asks some questions for the author to select some prompted choices. The author can provide a new file name. Typing "?" asks the system to assign a temporary file name and prompt it on the screen. If the typed file name is existing, a warning will appear. After a successful "create", the second menu will show.

The second entry "edit" is used to edit a voice data file. It takes an existing file name. Warning will given if the file does not exist. After a successful "edit", the second menu will appear on the screen.

The third entry "show" shows all the voice files with vos suffix in the local directory. The fourth entry "rename" renames an old file name in the local directory to a new file name. The fifth entry "remove" removes a voice file in the local directory. The sixth entry "exit" leaves the voice editor and returns to DOS.

7.4 The Second Menu

The second menu appears after "create" or "edit" in the first menu. The second menu is mainly for recording and playing a voice file. Currently, it has three entries described as follows. The first entry "record" is used to record voice in a file. And it can be interleaved with "play" before being "save"d. It takes one parameter for starting position in the voice file for recording. "0" is from file head, "1" (default) is from current position. and "2" is from file end. "record" is stopped by typing any key.

The second entry "play" is used to play some voice file. After the user selects "play", the system prompts and takes one integer n followed by a return as input, where n indicates starting playing from the n th block. For example, 0 means the beginning of the file. Such a feature replace the function of rewind.

The third entry "save" is used to terminate the recording and playing session and save it onto disk. After a successful "save", it returns to the first menu.

7.5 COMPILING AND LINKING COURSE (TRANSLATE)

It is invoked by typing "translate filename", where filename is a course file name with ".cor" dropped. Each .unit file is translated into a .ooo file if successful. And then all the .ooo files are linked together to resolve some cross references among the units in the course. When the link process is done successfully, a .run file is created with the same file name. If there are some syntax or linking errors, "translate" will prompt the error.

8. REHEARSING COURSE (REHEARSE)

This is invoked by typing "rehearse filename", where filename is a run file with ".run" dropped. It interprets the courseware and interact with users. If "-d" option is used, the error messages at interpretation time will be shown to the users.

9. MAKING COURSEWARE (EXTRACT)

This is invoked by typing "extract coursename". This tool will extract all referenced files in the coursename, such as the graphics files with .scr suffix and voice files with .vos suffix in the working directory, together with coursename.run to put them on a floppy disk for courseware distribution. Some copy protection may be available in latter version. It may also provide the option to put database files into the same floppy disk containing the courseware.

10. DELIVERING COURSEWARE (DELIVER)

In the delivering system, "deliver coursename" is the only command available, "deliver" interprets the courseware as "rehearse" except no system error messages will be reported.

APPENDIX 1

A. GRAPHICS BOARDS
   1. IBM Color/Graphics Adapter
   2. IBM Enhanced Graphics Adapter
   3. Hercules Graphics Card (Black/White)

B. PRINTERS AND PLOTTERS
   1. IBM Color Printer
   2. IBM PC Graphics Printer
   3. HP 7470A Plotter
   4. HP 7475A Plotter
   5. HP 2225C Printer
   6. DATA SOUTH 180 Printer
   7. DIABLO 150 Printer
   8. EPSON MX-100 Printer
   9. EPSON MX-80 with Graphtrax Plus Printer
   10. HOUSTON INSTRUMENT DMP-29 Plotter
   11. AMDEK AMPLOT II Plotter
   12. INTEGRAL DATA SYSTEMS Color Prism 80, 132
   13. INTEGRAL DATA SYSTEMS Black Prism 80, 132
   14. INTEGRAL DATA SYSTEMS Microprism 480
   15. NEC 7730 printer
   16. NEC 3550 Printer
   17. NICOLET ZETA 8 Plotter
   18. OKIDATA Microline 93, 94 with step 2 support
   19. OKIDATA Microline 92
   20. PRINTRONIX MVP, P300 and P600 printers
   21. QUME Sprint 11 Plus Printer
   22. SEIKOSHA GP-700A Color Printer
   23. STROBE Plotter

C. OTHER PERIPHERALS
   1. IBM Joystick
   2. Microsoft Mouse
   3. Mouse Systems PC Mouse
   4. GSS Metafile
   5. KODAK Pad Touch Tablet
   6. SUMMAGRAPHICS Summamouse
   7. SUMMAGRAPHICS Summatablet